2026
WINTER SCHOOL
in AI and Predictive Agriculture

Transformers

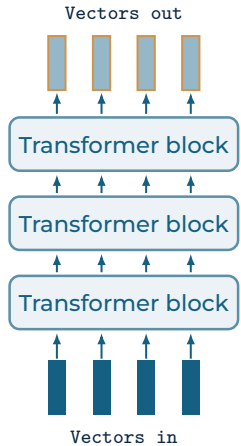
The architecture of the generative transformer

A/Prof. Yoni Nazarathy

The University of Queensland and
Accumulation Point Pty Ltd

The plan

1. **Everything is a vector**
2. **You need a little bit of mathematics**
3. **The attention mechanism**
4. **The transformer architecture**
5. **A survey of some architectures and specs**



Resources — if you want to go deeper

3Blue1Brown

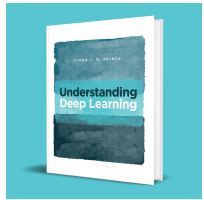
Beautiful visual intros to neural networks & LLMs

Mathematical Eng. of Deep Learning

Liquet, Moka & Nazarathy

transformers: Ch. 7

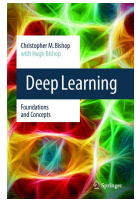
`deeplearningmath.org`



Understanding Deep Learning

Prince

transformers: Ch. 12



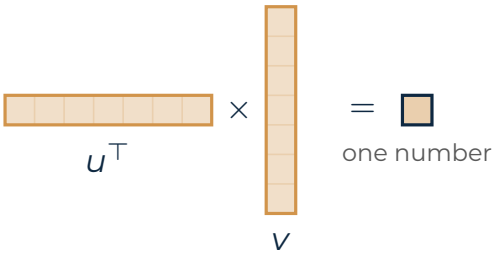
Deep Learning (Bishop)

Bishop & Bishop

transformers: Ch. 12

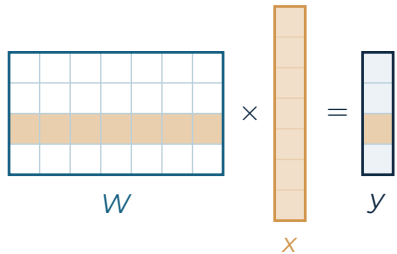
The two key operations

inner product



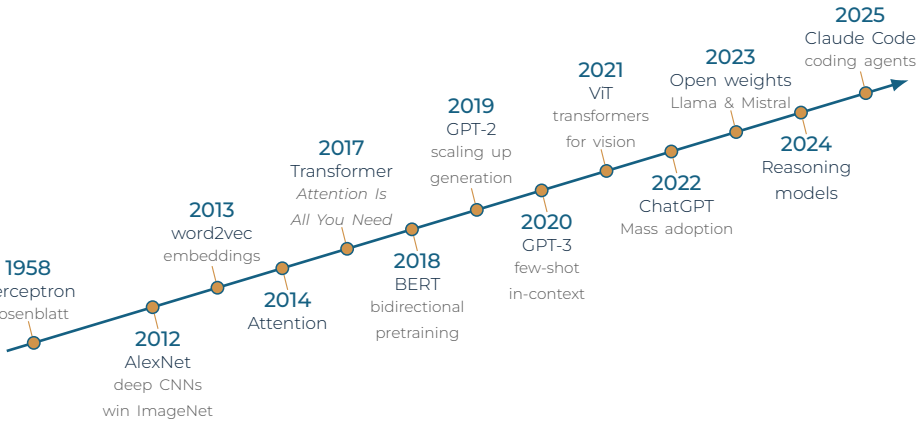
$$u \cdot v = u^\top v = \sum_j u_j v_j$$

matrix–vector product



$$y = Wx$$

Historical timeline



“Your” daily usage

- Every **token** = one **forward pass** through the whole network
- The pass is **matrix** × **vector**, over and over
- Each pass ≈ $2N$ operations ($N \approx$ **100 billion** parameters)

So let’s just **multiply it out** →

	~ 10	chats / day
×	$\sim 10^3$	tokens / chat
	~ 1.3 tokens per word · + prompt & history	
=	$\sim 10^4$	forward passes / day
×	$\sim 10^3$	matrix–vector products / pass
=	$\sim 10^7$	**matrix–vector products / day**
×	$\sim 10^8$	multiply-adds each
	$\approx \mathbf{10^{15}}$	**multiply-adds / day**

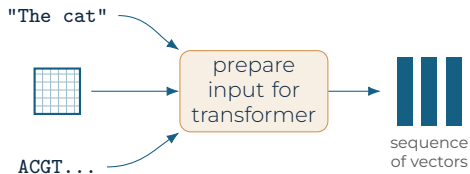
a quadrillion a day — and you don’t feel a thing

Part 1

Everything is a vector

The one thing a transformer eats

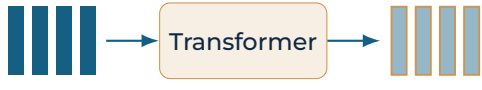
- A transformer only understands **a sequence of vectors**
  - text → **tokens** → vectors
  - image → **patches** → vectors
  - audio, DNA, …
- Once the input is a sequence of vectors, the **architecture is the same for any modality**



A transformer maps vectors to vectors

- A sequence of **vectors in** → a sequence of **vectors out**
- **Same length** — one vector per position
- Each output = its input, **rewritten in context**

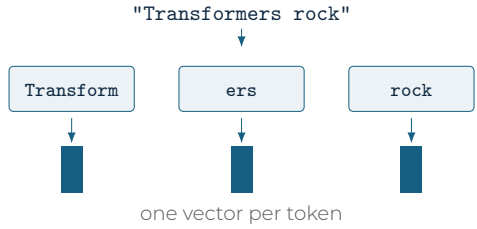
⇒ Those output vectors are then **read out for meaning** — generate text, classify, score, …



same length — each y_i is x_i, in context

Text: tokens, then vectors

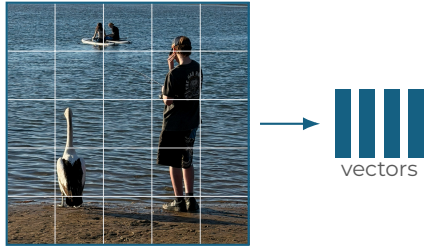
- Split the text into **tokens** (words / word-pieces)
- A fixed **vocabulary**; each token has an ID
- Look up each ID's **embedding vector**
- Result: **one vector per token**



Images: patches, then vectors

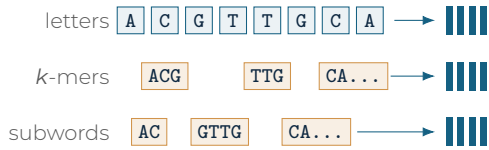
- Cut the image into a grid of **patches**
- Flatten each patch → a **vector**
- Read patches in order = a sequence

⇒ identical to text (this is **ViT**)



Biology: sequences into tokens

- **DNA** = a 4-letter language (A/C/G/T)
- **Protein** = a 20-letter language (amino acids)
- Same “input → tokens → vectors” step — just **swap the vocabulary**
- Tokenize by: single letters, ***k*-mers**, or **learned subwords**

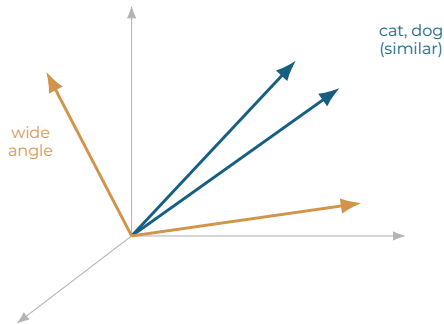


Meaning lives in the vectors

- The vectors aren't random — they're **learned**
- **Similar** meaning → **small angle** (close)
- **Unrelated** meaning → **wide angle** (far)
- Directions carry meaning (*king* − *man* + *woman* ≈ *queen*)
- The "embedding" from the previous talk

$$\texttt{cat:}\quad [0.21, -1.03, 0.44, \dots]$$

$$\texttt{dog:}\quad [0.19, -0.97, 0.50, \dots]$$



Context: the same word, different meaning

- e.g. “*river* ***bank***” vs “*money* ***bank***”
- Meaning depends on the **other words**
- Older models read left-to-right, one step at a time
- Long-range links get **forgotten**

⇒ we need a way to mix in **context**

Part 2

You need a little bit of mathematics

Matrix × vector — two views

- A matrix times a vector: Au
- **Row view:** each output entry is an **inner product** — a row · u
- **Column view:** the output is a **linear combination of the columns**
- Same answer — two lenses

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 3 & 0 \end{bmatrix}, \quad u = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

rows → inner products

$$\begin{bmatrix} 2 & 0 & 1 \end{bmatrix} u = 3$$

$$\begin{bmatrix} 1 & 3 & 0 \end{bmatrix} u = 7$$

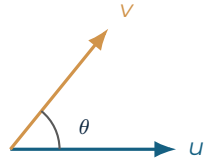
columns → linear combination

$$1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 3 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \end{bmatrix}$$

Inner product, angle, and room in high dimensions

- Inner product measures the **angle**: $\cos\theta = \dfrac{u \cdot v}{\|u\|\,\|v\|}$
- **Orthogonal** $(\theta = 90^\circ) \Rightarrow u \cdot v = 0$
- In d dimensions: exactly d orthogonal directions
- GPT-3 uses $d \approx$ **12,000** — so ∼12,000 truly orthogonal meanings

→ but allow **almost** orthogonal and **far** more fit

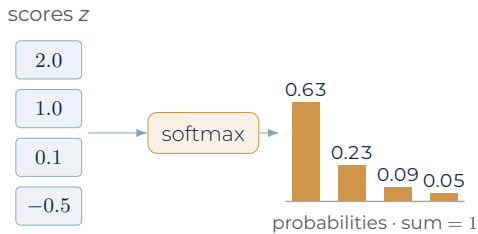


Softmax: numbers into probabilities

- Turns a **list of numbers** into **probabilities**

  $p_i = \dfrac{e^{z_i}}{\sum_j e^{z_j}}$ — all positive, sum to 1

- Bigger input → bigger share **(exponentially)**
- We'll meet it twice: **attention weights** and **next-token** choice

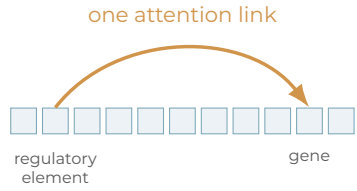


Part 3

The attention mechanism

Attention = long-range dependencies

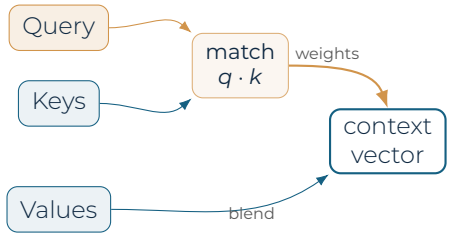
- Attention links elements **however far apart** they sit
- Biology is **full of long-range links**:
  - a regulatory element sits **thousands of bases** from its gene
  - amino acids far apart in the chain **touch once the protein folds**



Query, Key, Value — the three roles

- Attention works like a **search**:
- **Query** — what a word is looking for
- **Key** — what each word advertises
- **Value** — what each word passes on

→ match query to keys, then **blend the values**



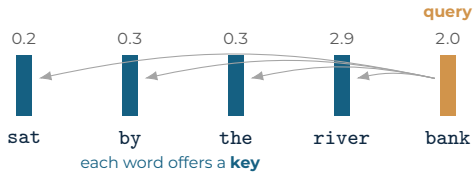
Self-attention: one word gathers context

each word starts as a vector

Self-attention: one word gathers context

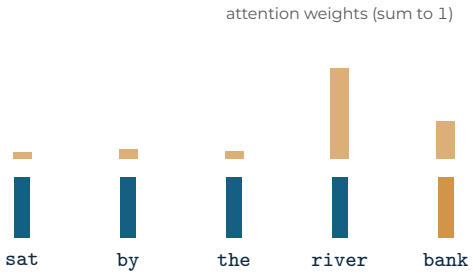
“bank” forms a **query** — *what around me matters?*

Self-attention: one word gathers context



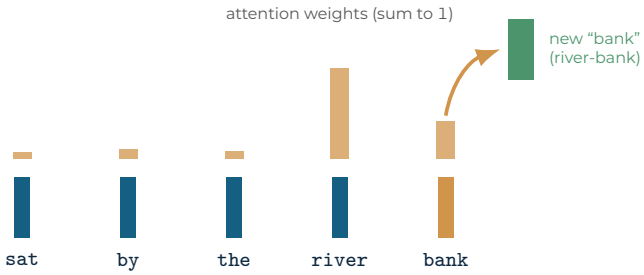
score every word: query · key

Self-attention: one word gathers context



softmax → weights — “bank” mostly hears “river”

Self-attention: one word gathers context

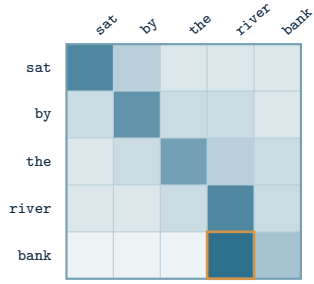


blend the values → a context-aware “bank”

… and every word does this at once

- Every word runs its own search **in parallel**
- Each row = one word's **attention weights**
- Brighter = **more attention**

→ out comes a **new sequence of vectors**



Attention in symbols — simpler than it looks

- **Score** two vectors by their **dot product**
- **Softmax** the scores → weights (sum to 1)
- Output = **weighted blend** of the vectors
- **Learned parameter matrices** W_Q, W_K, W_V give the query, key, value

1. compute query, key, value

$$q^i = W_Q\, x^i \quad k^j = W_K\, x^j \quad v^j = W_V\, x^j$$

2. attention weights with softmax

$$\alpha_{ij} = \mathrm{softmax}_j\left(\frac{q^i \cdot k^j}{\sqrt{d}}\right)$$

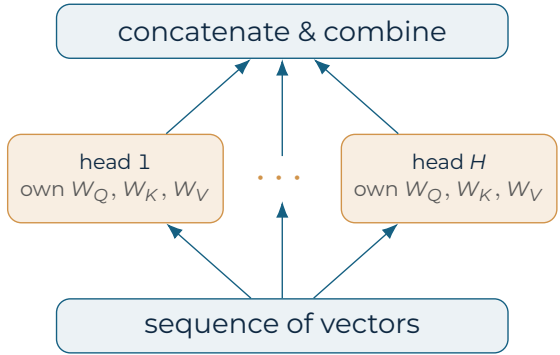
one α_{ij} per pair

3. combine values by those weights

$$y^i = \sum_j \alpha_{ij}\, v^j$$

Multi-head attention

- Run attention **several times in parallel**
- Each “head” can focus on a **different pattern** (grammar, subject, long-range links…)
- Concatenate the heads → combine
- More views of the same sequence

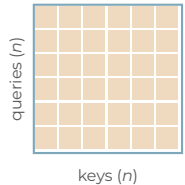


Why attention costs $O(n^2)$

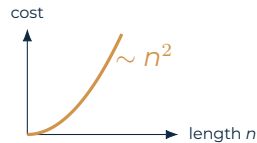
- Every vector attends to **every other vector**
- n vectors $\Rightarrow$ an $n \times n$ **table** of query $\cdot$ key scores
- Softmax & blend run over **all n^2 of them**

$\Rightarrow$ cost grows **quadratically** with the length n

- Long sequences (books, genomes) get **expensive fast**



$n \times n = n^2$ scores

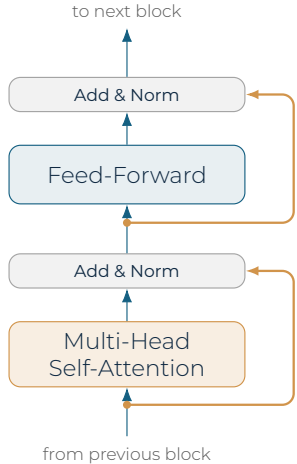


Part 4

The transformer architecture

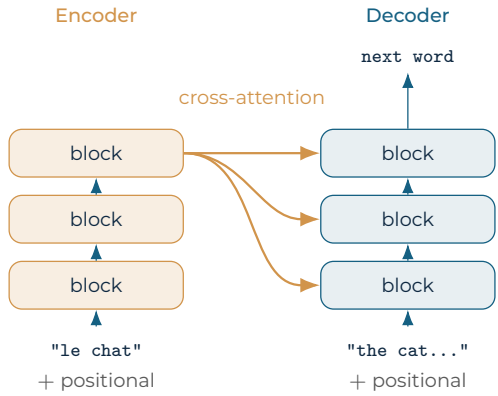
The transformer block

- Two sub-layers:
  - **Multi-head self-attention** — mix context
  - **Feed-forward network** — think per token
- **Residual + Layer Norm** keep training stable
- Same block **repeated** many times



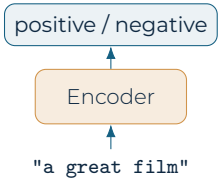
Stacking blocks: encoder & decoder

- Stack blocks into deep towers
- **Encoder** — understands the input
- **Decoder** — generates the output
- GPT-style LLMs are **decoder-only**
- **Positional embeddings** put order back in



Three variants — same block, wired three ways

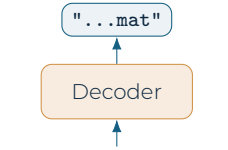
Encoder-only



e.g. BERT

understand & classify

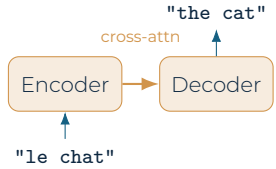
Decoder-only



e.g. GPT / LLMs

generate text

Encoder–decoder



e.g. T5

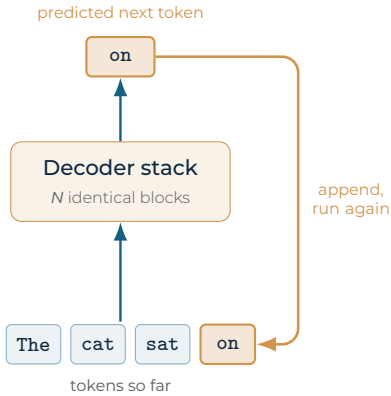
read one, write another

Similar transformer blocks inside — in all three cases; only the wiring and training differ.

An LLM is decoder-only

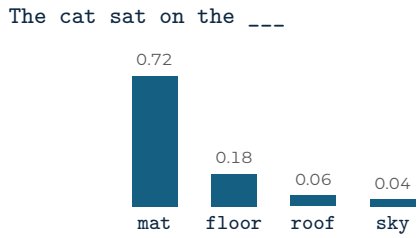
- Trained from scratch to **just continue text**

It writes **one token at a time** — each new token is **appended and fed back in** (*autoregression*).



Language modelling: predict the next token

- Task: given the text so far, **predict what comes next**
- Output is a **probability** over the whole vocabulary
- Pick one, append it, **repeat**
- That loop *is* text generation



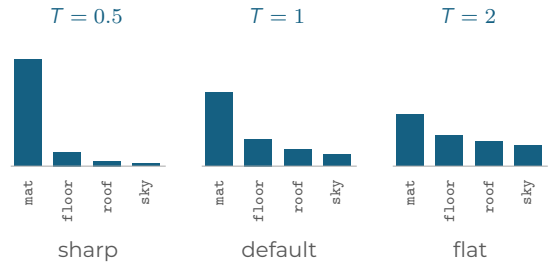
Temperature & sampling: choosing a token

- Divide by **temperature** T before softmax:

$$p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

- **Low** T — sharp / peaked; **high** T — flat / spread out
- Same numbers, a **dial** between confident and random

→ then **pick** a token: **greedy** or **sample** (**top-**k / **top-**p)



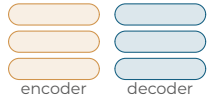
Part 5

A survey of some architectures and specs

The original Transformer — 2017

- *Attention Is All You Need*
- Dropped recurrence — **attention only**
- Built for **machine translation**
- Introduced **scaled dot-product** & **multi-head** attention

Vaswani et al., *NeurIPS* 2017



Shape	encoder–decoder
Params	65M / 213M
Layers	$6 + 6$
Heads	8, $d = 512$
Data	WMT translation

BERT — 2018: the understanding engine

- **Encoder-only** — reads **both directions** at once
- Pretrain by **masking** words and filling them back in
- Fine-tune for classification, QA, tagging
- Perhaps the template for today's **biology encoders**

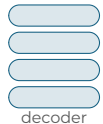
Devlin et al., *NAACL* 2019

Shape	encoder-only
Params	110M / 340M
Layers	12 / 24
Pretrain	mask words (MLM)
Data	Wiki + books, 3.3B words

GPT-2 — 2019: just predict the next word

- **Decoder-only** generator
- One job: **predict the next token**, at scale
- First **coherent long-form** text
- Surprising **zero-shot** abilities — no fine-tuning

Radford et al., OpenAI 2019



Shape	decoder-only
Params	up to 1.5B
Layers	48
Context	1024 tokens
Data	WebText, ~40GB

GPT-3 — 2020: scale unlocks few-shot

- Same recipe as GPT-2, ~**100×** **bigger**
- **In-context learning** emerged — show examples in the prompt
- No fine-tuning needed — just **ask**
- The base that became **ChatGPT**

Brown et al., *NeurIPS* 2020

Shape	decoder-only
Params	175B
Layers	96, $d = 12288$
Context	2048 tokens
Data	~300B tokens

State of the art — open-weight models

- Open weights now **rival closed frontier models**
- **Llama** (Meta), **DeepSeek**, **Qwen**, **Mistral**
- **Mixture-of-experts**: huge, but only a slice runs per token
- **Long context** & **built-in reasoning**

→ download from **Hugging Face** 🤗, run locally, fine-tune

Llama 4 (Meta, 2025)
MoE 17B active (Scout / Maverick)

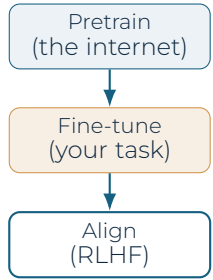
DeepSeek-V3 / R1 (2025)
MoE 671B total, 37B active

Qwen3 (Alibaba, 2025)
MoE 235B total, 22B active

trained on 10–15T tokens · 128k+ ctx

Pretraining then fine-tuning

- **Pretrain** — predict the next word on *huge* text
  general language ability, no labels needed
- **Fine-tune** — a little targeted data for a task
- **Instruction tuning + RLHF** — learn to be helpful
  this is what turned GPT-3 into ChatGPT



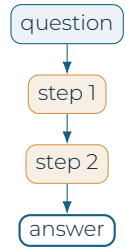
In-context learning

- Show a **few examples in the prompt**
- Model adapts — **no re-training**
- “Few-shot” behaviour appeared with GPT-3
- Prompting becomes a **skill of its own**

```
sea → mer
sky → ciel
cat → ?
```

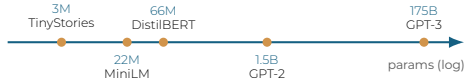
Reasoning (“thinking”) models

- Let the model **think step by step** before answering
- Spend **more compute at answer time**
- Trained to produce useful **chains of thought**
- Big gains on maths, code, planning (`o1`, DeepSeek-`R1`, …)



Small Transformers?

- Unlike simpler deep learning — **no clear evidence** that *ultra* small transformers work well
- Bigger is predictably better — but you don't need billions
- **TinyStories**: a ~**few-million**-param model writes coherent little stories
- **MiniLM** (~22M): workhorse behind **semantic search / RAG**
- Distilled encoders (**DistilBERT**, 66M) run on a **laptop or phone**



Thank you

with thanks to our partners

ARC CENTRE OF EXCELLENCE FOR PLANT SUCCESS IN NATURE AND AGRICULTURE

appn Australian Plant Phenomics Network

AAGI Analytics for the Australian Grains Industry

ARC Training Centre in Predictive Breeding

QUT Queensland University of Technology

The University Of Queensland AUSTRALIA

QAAFI